Software Engineering and Testing. BSC Year 2, 2020/2021
(Assignment 3 - 20%)

Assessment 3: Design and Draft Implementation

Submitted by:

David Thornton B00152842

Patryk Miciniak B00154442

Alexandru Diaconu B00151494

Submission date

18/03/2024

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ordinary Degree in Computing in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated.

Author: Patryk Miciniak Dated: 18/03/2024

Author: Alexandru Diaconu Dated: 18/03/2024

Author: David Thornton Dated: 18/03/2024

Table of Contents

Contents

1. Title: GymGo

2. Abstract / Executive Summary

This document outlines the general development of GymGo which is a software system designed to manage gym operations efficiently. The system has many useful features such as booking lessons, user registrations with login/logout functions, purchasing products from an online store and cancelling bookings it is mainly aimed to improve the users experience and help manage the gym better.

The development follows a structured methodology it uses Unified Modelling Language (UML) for system modelling and Object-Oriented Analysis and Design (OOAD) to ensure scalability, flexibility and maintainability. These methodologies help split big programs into smaller more manageable parts and it enables easier maintenance and future scalability.

Throughout the project the focus has been placed mainly on security measures, user interface simplicity and easy navigation to make the software more user friendly and more secure. The initial goals have been met with big improvements made based on the main considerations such as implementing enhanced security features to help protect user data.

In conclusion GymGo represents a significant advancement in gym management simplifying processes while protecting user data and enhancing the overall user experience.

3. Project Definitions

- Purpose of document

  The purpose of this document is to outline the functionalities and processes involved in the development of a software system designed for managing bookings, registrations, logins/logout, purchasing products, and cancelling bookings within a gym or fitness centre. It provides a structured guide for developers, stakeholders, and users to understand how the system operates and its key features.

- What is the project?

  The project involves the development of a comprehensive software system tailored for gym or fitness center management. It encompasses features such as booking lessons, user registrations, login/logout mechanisms, purchasing products from an online store, and cancelling bookings. The system aims to streamline administrative tasks, enhance user experience, and facilitate efficient management of resources within the fitness facility.

- Functional Specifications

1. Booking Lessons: Users can select a desired day to book a lesson. The system retrieves available lessons for that day from the database, displays them to the user, and allows them to make a booking. It also checks for any existing bookings by the user at the same time and provides options accordingly.

2. Registration: Users are required to provide personal details such as name, surname, age, address, phone number, email, and create a unique password for registration. Upon submission, the system sends the registration details to the database, which saves them for future logins using email and password authentication.

3. Login/Logout: Authentication is required for users to access the system. Users provide their credentials (email and password) for login, and the system verifies them with the database. Upon successful authentication, users gain access to the system's workspace. Administrators have a separate login process with username and password authentication.

4. Online Store (Buying): Users can browse through available products in the online store, view detailed information about specific products, and proceed to purchase them. The system retrieves product information from the database,

processes purchase requests, manages inventory levels, generates transaction records, and provides confirmation messages upon successful purchases.

5. Cancelling Lessons: Users can view their booked lessons, select a booking for cancellation, and confirm the cancellation. The system retrieves booked lessons from the database, processes cancellation requests, and updates the user's booking status accordingly.

- Main components of the software system

- Logging in/out - Registering

- Booking lessons - Cancelling lessons

- Purchasing products - Paying

4. Document Revision

Rev. 1.0 15/03/2024 – initial version

Rev. 2.0 17/03/2024 – points outlined

Rev. 3.0 18/03/2024 – points expanded

5. Methodology

In the creation of this web application, several methodologies were used in the design and implementation. System models, based on the unified modelling language, where used on when developing the application to ensure that they carry out all necessary functions with efficiency and as intended.

The application was made with an object-oriented approach. This was done to ensure the project could easily be upscaled and that components of the project could be reused. For instance, if the application was to later be expanded to include more features/components, properties from current classes could be reused in new classes (with generalization). Using classes also made it easier for components to interact with both current components and possible new components with use of aggregation and composition. For instance, the products section of the application could easily be upscaled with new product types by using the current product class to act as a parent class to different subtypes. Alternatively, if new classes are created which also have images, the current image class can be used for the new classes image information. To aid in the design of the application and its classes, class diagrams, static use case diagrams which show classes with their methods and multiplicities, were used.

Class diagrams were used to aid in the planning and implementation of classes and to aid in database design. Class diagrams were used to identify what components of the application required persistent/permanent storage and which components only needed temporary storage while the application was running. For persistent storage a database was used, which was first designed with entity relationship diagrams, another type of static diagram, which shows data and its relationship. This diagram was used in planning the database to ensure that the database was as efficient as possible and in the 3rd normal form, without repeating information or variables with multiple values.

Although a database is needed for persistent storage, it is impractical to read and write information back and forward to the database repeatedly throughout the application and should be avoided if possible. This is an additional reason why all database information was stored in class objects during the running of the application. With this system modelling approach, static classes enable the users to dynamically interact with objects within the program without writing back and forth to the database.

Users can interact with these objects using the user interface, which has been dynamically designed to allow users to easily interact with the application. The user interface has been developed with user interface principals in mind, most notably ensuring the application is responsive, simple and consistent. This includes ensuring the interface provides relevant feedback to users when interacting with the application and that the interface was designed with simplicity and consistency in mind to ensure the system is learnable and simple to use.

All of these methodologies and system modelling approaches have been implemented to ensure the application not only fulfils its function but is efficient in doing so while still being user-friendly.

6. Requirements

6.1 Use Case Specifications

Book Lesson:	• Users will select a day they want to book a lesson. • System makes a request to search for all available lessons that day. • The database returns the lessons of that day. • The list of lessons is displayed to the user. • The user selects to book the lesson they want. • The system checks if they are logged in and instructs them to if they are not (See login/logout).

	• The system makes a request to see if the user has booked any other lessons at this time. • The database returns a lesson at that time or nothing. • If the user has a lesson booked at this time they are warned and provided the option to cancel or continue. • If the user continues, the system sends a message to the database to update who is attending the lesson. • The system displays a confirmation message to the user and gives them the option to return to the home page or continue in the booking page. • The user is directed to whichever page they select.
Register:	• Users must put their name, surname, age, address, phone number, email address and to create a unique password. • The user than clicks the submit button. • System sends all the registration details to the database. • Database return the Login successful details to the system. • The database saves all the user registration details so the next time they login is by using their email and created password. • System confirms user with a message - successful registration.
Login/Logout:	• Administrator must login to the system using his username and password. • Authentication request sent to the database to confirm administrator`s details. • Authentication verified. • Workspace displayed to the administrator
Browse Products in Online Store:	• The user selects the Online Store section. • The system makes a request to retrieve all the items for sale from the database. • The database returns the items available for purchase. • The system displays the items for sale to the user.
Manage Shopping Cart	• User adds products to their shopping cart while browsing the Online Store section. • System updates the shopping cart with the selected products. • If the user adds a product: • System checks if the product already exists in the cart. • If yes, system increments the product quantity. • If not, system adds the product to the cart. • If the user updates a product: • System updates the quantity of the specified product in the cart. • If quantity of the specified product is 0: • System removes the specified product from the cart.
Checkout and Place Order	• User fills out shipping and payment details on the checkout page.

	• System validates the form data for correctness. • If the form data is valid: • System submits the order for processing. • For each product in the shopping cart: • System records the order details including user ID, product ID, quantity, and order time. • System clears the shopping cart. • System redirects the user to the order confirmation page. • If the form data is invalid: • System displays error messages to the user indicating the validation errors. •
View Order Confirmation	• User is redirected to the order confirmation page after placing an order. • System retrieves the order details for the user from the database. • If order details are found: • System displays a thank you message to the user for their order. • System displays the order details including the products purchased and their quantities. • If no order details are found: • System notifies the user that order details could not be found and prompts them to ensure their order was processed correctly. • User can continue shopping by clicking the "Continue Shopping" link provided on the page.
Cancelling Bookings	• The user selects the booked section of the lessons page. • The system checks if they are logged in and instructs them to if they are not (See login/logout). • The system makes a request to see all the lessons booked buy that user. • The database returns the booked lessons of the user. • The system displays the lessons booked by the user. • The user then selects to cancel the booking. • The system sends a warning to the user and asks if they want to continue. • If the user continues, a message is sent to the database to delete that lesson booking. • The system informs the user that their booking has been cancelled. • The system displays the booked lessons of the user, without the deleted booking.

6.2 Classes and methods created from Use Case specification.

The decision to identify User, Admin, and Customer entities, coupled with inheritance implementation, aimed to streamline code management and promote scalability. By consolidating shared attributes and functionalities into a base class (User), redundancy was minimized, enhancing code reusability and maintainability. This architectural approach ensures seamless adaptation to future user type expansions while facilitating consistent handling through encapsulation and polymorphism. Overall, it optimizes code organization, simplifies maintenance, and lays a robust foundation for future system growth.

The book lesson section highlights different lessons the user can book. The information regarding these lessons can be stored in a class. This information must first be retrieved from the database, so a method will be used for this. The use case also describes this information being displayed to the user to view before booking a lesson. A method can be used to display this information in the required format.

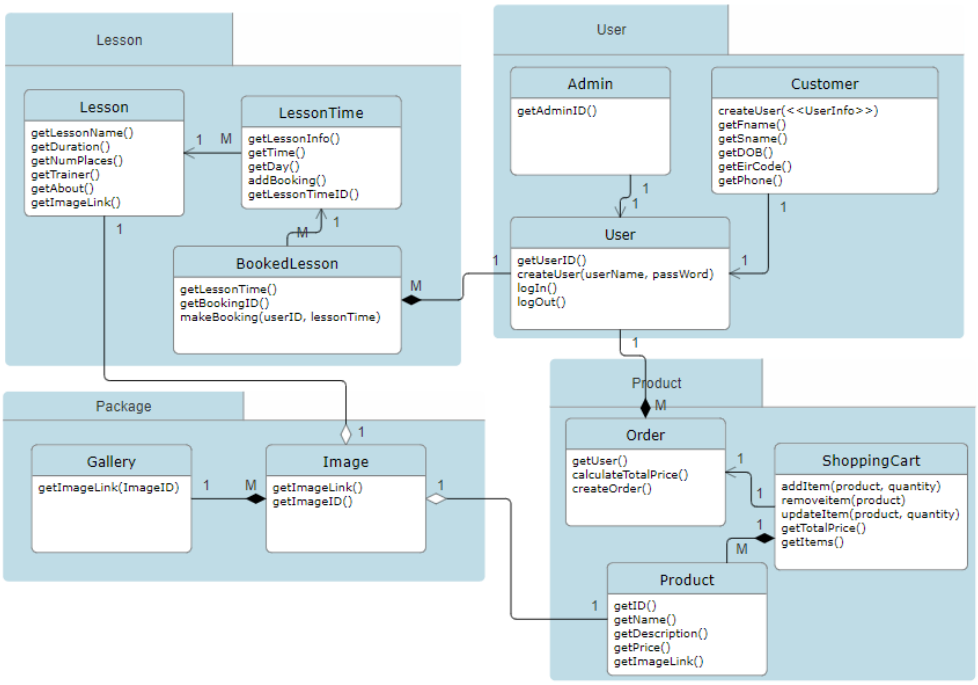
The use case also clarifies that different lessons can be booked at different times on different days. To sore data and time information a second lesson class called lesson time will extend from the original lesson class, a method to retrieve this information from the database will also be needed. The main functionality of the lessons section is that the users can book lessons, provided they are logged in, with the lesson information, time and date. To associate this information with the user, a booked lesson class can be created to store this information, with relevant methods for users to make the booking and to send the booking to the database. The user can also view any lessons they have previously booked as described in the cancel booking section. These bookings will be retrieved from the database with a method. Another method will be used to delete the bookings from the database if the user selects the delete button as described in the use case.

In the online store section of GymGo users can explore a variety of products available for purchase. The system upon entry into this section, queries the database to fetch all items for sale then displays them for the user, products chosen by the users are added to a virtual shopping cart. If a product is already present its quantity is updated rather than duplicated.

When ready users proceed to check out, they fill out necessary shipping and payment details. The system validates this information, processes the order by recording the transaction details including user ID and product specifics, then clears the cart for future transactions. Upon successful order placement the users are redirected to a confirmation page which reassures them of their successful purchase by listing order details or if there was a problem it prompts them with the error code.

7. Case Diagrams

7.1 Class Diagram



All classes have a constructor to create the object with information from the database and getters to retrieve that information for later use. Some classes have extra methods which will be outline below.
The application will feature a gallery the user can view from one of the pages. This gallery has a class to store images to display with a method to display the image name. This class will be made up of multiple objects from an image class. As a gallery will need an image to function, a full aggregation was made between them.

In the decision-making process, three primary entities were identified: User, Admin, and Customer. An important step was the creation of a user class as the foundational base, housing essential attributes and methods shared by both Admin and Customer entities. By implementing inheritance, Admin and Customer classes inherit these fundamental features from the User class, facilitating code reusability. This strategic approach not only reduces redundancy but also enhances maintainability by centralizing updates to common functionalities. Scalability is seamlessly achieved

through the flexibility of inheritance, enabling effortless integration of new user types. Encapsulation further organizes related properties and methods within the base class, optimizing code management and organization. Moreover, the application of polymorphism empowers subclasses such as Admin and Customer to be treated uniformly as instances of the base User class, ensuring consistent and versatile handling throughout the system.

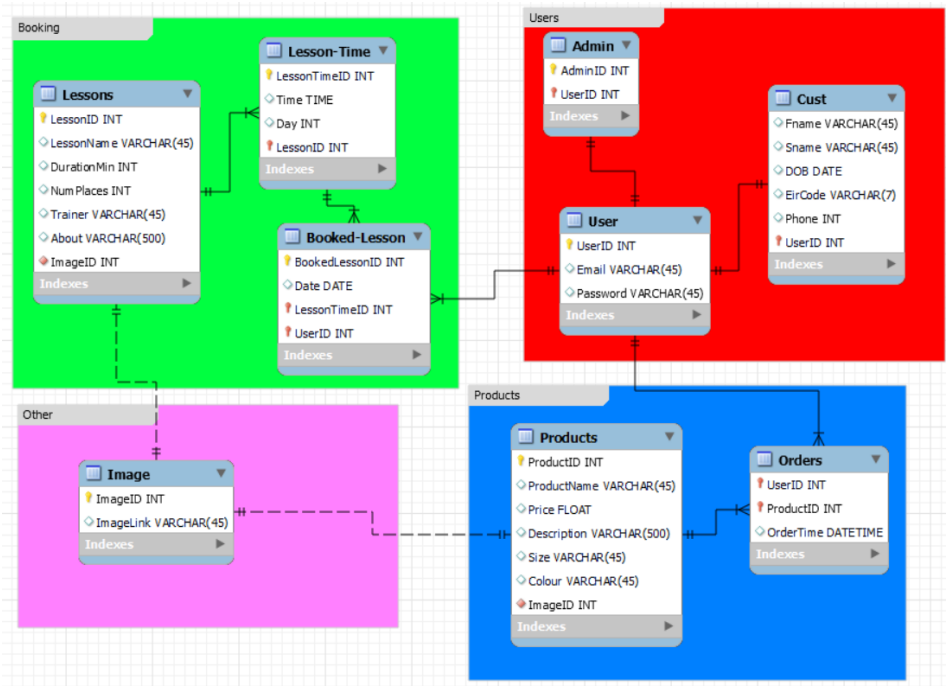
A lessons class was created to store information regarding lesson for user to view. As each lesson has multiple times a child class was made from lessons called lesson time to store time information. This class was extended from lessons so that it could also use the properties of lessons. Users have the option to book multiple lessons. Too associate lessons with users and to keep relevant booking information a booked lesson class was made. This inherits form lesson time so it can use variables from the related lesson and has a full aggregation with users as without a user the booking wouldn't exist. As a booking can either be made by the user input or gotten from the database booking has an additional method to create the booking with different variables.

Each lesson can have one image associated with it; however, lessons can still be displayed without an image so a partial dependency was created between image and lesson. Along with lessons, products can also have an image associated with is but also don't need an image to be displayed, and hence also has a partial dependence with the image class.

The Product class is designed to encapsulate essential details for product listings such as ID, name, description, price and an optional image link, this is a partial dependency with the Image class, indicating products may have images to enhance listings but aren't dependent on them. To manage user interactions with these products particularly the addition of items to a shopping cart the Cart class is used. This class dynamically updates to reflect user selections, offering methods to add, update, or remove products. This setup demonstrates dependencies where the Carts functionality is directly reliant on Product details. The checkout process, managed by the submit order method, validates user information and records order details in the database associating each order with the users ID, product IDs, and quantities showcasing an aggregation relationship between User and Order classes. An orders existence is aggregated by user actions. After placing an order, users are directed to an order confirmation page where the get order details for user method retrieves and displays a summary of their purchase.

7.2 Entity Relationship Diagram

Entity Identification:



Establishing a 1 to 1 connection from Customer to User: Each customer record is associated with exactly one user record, ensuring registration details are stored securely for authentication during login.

Establishing a 1 to 1 connection from Admin to User: Each administrator record is associated with exactly one user record, simplifying the authentication process by using the same mechanism as for regular users.

Overall, the user specification facilitated the structuring of the database and ensured consistent and effective authentication mechanisms for both customers and administrators.

Each lesson has lesson information specific to each lesson along with multiple times and days the lessons are at. To avoid repeating values in rows, a second table for lessons, lesson-time was created. Each lesson can have multiple users booking the lesson and each user can book multiple lessons, to sore the booked lessons with relevant extra information and without variables with multiple values (i.e. multiple user ids in each lesson time) a booked lesson table was created. Each lesson or product can have one image

associated with it, and each image that is associated with a lesson or product can only be associated with one. However, as each lesson/product doesn't need an image, a partial dependency has been created between them.

Each product is detailed in the Products table with unique information such as ID, name, price, description, size, colour and an optional link to an image, ensuring comprehensive data without redundancy. The design incorporates a Products table to hold product attributes, preventing repetition and ensuring data normalization.
To handle transactions the Orders table is introduced, connecting users to their purchased products. This table records each order by associating it with a users ID, product ID, quantity and order time it demonstrates a many to many relationship between users and products through orders. This structure captures the fluidity of transactions where users can purchase multiple items and items can be bought by multiple users.
The Image table associated with products through a foreign key ImageID illustrates a one-to-one relationship, signifying that while each product may be linked to a single image to enhance its presentation, the association is not mandatory. This partial dependency between Products and Image acknowledges that products can exist without images this allows for flexibility in product listings.
Through these relationships and table structures the database design adeptly supports the systems needs for product listing, order processing, and optional product visualization which ensures an organized, scalable and user-friendly shopping experience.

8. Conclusions

The GymGo project has made significant progress since its inception to where it stands now. Originally designed to simplify gym management. We added important features such as signing up users, scheduling classes, buying products and cancelling bookings which were all in alignment with our original objectives.

However, as the project evolved practical considerations led to several necessary changes. It became crucial to implement improved security measures. We adopted the escape PHP function to showcase our dedication to security. This function sanitizes user input to prevent Cross Site Scripting (XSS) attacks by converting special characters to HTML entities it eliminates unnecessary whitespaces and removes backslashes used to escape quotes. Such measures ensure that our platform not only serves its intended purpose but also protects the integrity and privacy of our user's data.

Furthermore, the design of the project was improved to make it easier for users to move around and interact with the platform. This included making the user interface simpler and rethinking the way users move through the site. These changes made the platform much more user friendly.